SwinxsTalk Reference Card – Expressions & Statements

Syntax	Type	Meaning
<constant>	expression	An integer or a constant defined in a declaration-block.
<var>	expression	Retrieves the value of <variable>.
<var>[<expr>]	expression	Retrieves from the array <var> the value indexed by <expr>. Indexes should run from 0 up to, but not including, the size of the array.
<expr> + <expr>	expression	Addition.
<expr> - <expr>	expression	Subtraction.
<expr> * <expr>	expression	Multiplication.
<expr> / <expr>	expression	Integer division.
<expr> % <expr>	expression	Modulo.
<test> ? <expr> : <expr>	expression	If <test> is true, it equals the first <expr>, otherwise the second.
min(<expr>, <expr>)	expression	Minimum.
max(<expr>, <expr>)	expression	Maximum.
random(<expr>)	expression	Returns a random value between 0 and (<expr> - 1).
random(<lo>, <hi>)	expression	Returns a random value between the expressions <lo> and <hi>.
<expr> && <expr>	boolean	True only if both arguments are true.
<expr> || <expr>	boolean	True only if one or both of the arguments are true.
! <expr>	boolean	True only if the argument is not true.
<test> ? <expr> : <expr>	boolean	If <test> is true, it equals the first <expr>, otherwise the second.
<expr> == <expr>	boolean	Equality test on a variable.
<expr> != <expr>	boolean	Inequality test on a variable.
<expr> < <expr>	boolean	Test whether a variable is lesser than an <expr>.
<expr> <= <expr>	boolean	Test whether a variable is lesser or equal to an <expr>.
<expr> > <expr>	boolean	Test whether a variable is bigger than an <expr>.
<expr> >= <expr>	boolean	Test whether a variable is bigger or equal to an <expr>.
state.move(<expr>)	statement	The state machine will jump to state <expr> at the next cycle.
machine.state.move(<var>, <expr>)	statement	The state machine <var> will jump to state <expr> at the next cycle.
timer.on(<value>, <expr>)	statement	Start running timer <value> from <expr> and up.
timer.set(<value>, <expr>)	statement	Set timer <value> to <expr>.
timer.off(<value>)	statement	Pause timer <value>.

Syntax	Type	Meaning
timer.on(<value>)	statement	Resume timer <value>.
timer.get(<value>)	expression	Retrieves the current time of timer <value>.
timer.running(<value>)	boolean	True while timer <value> is ticking.
clock.gettime(<hour>, <minute>)	statement	Fills the word-vars with the current hour (0 to 23), and minutes (0 to 59) respectively.
clock.getdate(<year>, <month>, <day>, <dayofweek>)	statement	Fills the word-vars with the current year (2008 and up), month (1 to 12), day (1 to 31) and the day of the week (0 to 6, where 0 stands for Sunday) respectively.
rfid.on	statement	Turn on the detection of tags.
rfid.off	statement	Turn RFID detection off, e.g. to save energy.
rfid.detected	boolean	True for one cycle when a tag is detected. Fills rfid.last.lo and rfid.last.hi.
rfid.last.lo	expression	Return the RFID number modulo 65350.
rfid.last.hi	expression	Return the RFID number divided by 65350.
rfid.accessory.detected	boolean	True for one cycle when a accessory tag is detected.
rfid.xs.register.on	statement	Start a new registration procedure. Typically used in the game selection program.
rfid.xs.register.off	statement	Terminates the registration procedure.
rfid.xs.count	expression	Returns the number of registered Xses.
rfid.xs.detected	boolean	True (for one cycle) when a registered XS is detected. Stores the index of the XS in rfid.xs.last.
rfid.xs.last	expression	Returns the index of the current XS.
score.set(<XS>, <expr>)	statement	Stores expression <XS> in rfid.xs.last and <expr> in score.last. Saves the next cycle the score for this XS in the internal score table.
score.get(<expr>)	expression	Returns the score of the XS <expr>. *Valid only for <expr> from 0 up to 9 (e.g. the first 10 players).*
score.get(<expr>)	statement	Stores XS <expr> in rfid.xs.last and, at the next cycle, fills score.last (from the score table), rfid.last.lo and rfid.last.hi.
score.last	expression	Returns the score of rfid.xs.last.
score.saveall	statement	Write the score table of all registered XSes to the persistent memory. When the user connects his/her Swinxs the the PC, these scores are sent to swinxs.com.
sound.play(<type>, <expr>)	statement	Plays audio file <expr> of type <type>.
sound.chain(<expr>, <chain>)	statement	Plays a sequence of foreground sound samples. After starting the last sample, jumps to state <expr>. This command will commence as soon as no sample is playing.

SwinxsTalk Reference Card – Expressions & Statements

Syntax	Type	Meaning
sound.off	statement	Aborts playing a sample.
sound.playing	boolean	True while sound is playing.
sound.playing.pause	statement	Pauses playing (or recording) audio.
sound.playing.resume	statement	Resume playing (or recording) audio.
sound.playing.paused	boolean	True if foreground audio is paused.
sound.volume	expression	Gets the current foreground volume.
sound.volume(<expr>)	statement	Sets the foreground volume (from 45 to 100).
sound.visual.on	statement	Normally, when Swinxs speaks (plays audio), the LED pulses green. You enable this with sound.visual.on.
sound.visual.off	statement	Stops pulsing the LED while Swinxs speaks, e.g. if you want to program the LED yourself.
sound.background.play (<type>, <expr>)	statement	Plays background audio file <expr> of type <type>.
sound.background.off	statement	Aborts playing a background sample.
sound.background.playing	boolean	True while background sound is playing.
sound.background .playing.pause	statement	Pauses playing (or recording) audio.
sound.background .playing.resume	statement	Resume playing (or recording) audio.
sound.background .playing.paused	boolean	True if background audio is paused.
sound.background.volume (<expr_bg>, <expr_fg>)	statement	Sets the background audio volume. If foreground audio is playing, use <expr_fg>, otherwise <expr_bg>. Ranges from 0 (silence) to 100 (same level as current foreground sound).
sound.background.fade (<expr_bg>, <expr_fg>, <step>)	statement	Gradually change background volume settings. The background sound will abort when faded to a volume of 0. Each control cycle the volume gets in- or decremented by <step>. Typically you'll apply a step between 1 (slow) and 10 (fast).
sound.check(<type>, <expr>)	statement	Checks whether the audio file <expr> of <type> exists. You must test sound.exists the next control cycle.
sound.exists	boolean	True if the file specified by sound.check() exists.
sound.record.on(<expr>)	statement	Start a voice recording of a sound sample. <expr> holds the id of the sample in the game directory.
sound.record.add(<expr>)	statement	Start a voice recording of a sound sample. The first free (non used) sample from <expr> or up will be used as the sample id.
sound.record.xs(<expr>)	statement	Start a voice recording for the registered XS <expr> (0 to number of registered players).

Syntax	Type	Meaning
sound.record.off	statement	Finish the sound recording.
sound.detected	boolean	True when noise is detected (kids yelling/clapping their hands).
sound.remove(<expr>)	statement	Removes sample <expr>.
sound.remove.xs(<expr>)	statement	Removes a voice recording for the registered XS <expr>.
sound.jukebox.shuffle.on (<directory>)	statement	Initializes the jukebox using <directory>. Songs are shuffled.
sound.jukebox.sort.on (<directory>)	statement	Initializes the jukebox using <directory>. Songs are sorted by number.
sound.jukebox.next	statement	Moves to the next song.
sound.jukebox.previous	statement	Moves to the previous song.
sound.jukebox.count	expression	Number of songs in the jukebox directory.
sound.jukebox.off	statement	Disables the current jukebox. The next control cycle you could open another jukebox.
led.hsv(<h>, <s>, <v>)	statement	Set the LED to hue (<h> % 360), saturation (<s> % 101), and value (<v> % 101).
led.color(<color>, <int>)	statement	Set the LED to <color> and intensity <int>. The intensity should be “HIGH”, “MEDIUM”, or “LOW”.
led.rfid(<expr>, <int>)	statement	<expr> should be an RFID number modulo 65350 and <int> should be an intensity specification.
led.pulse(<color>, <speed>)	statement	Make the LED pulse in <color> with a duration of <speed>.
led.off	statement	Makes the LED black (turn off).
accel.move.detected	boolean	True when Swinxs is (slightly) moved.
accel.shake.detected	boolean	True when Swinxs is shaken.
accel.move.settings(<threshold>, <count>)	expression	Threshold (default 10) and count (default 4) settings defines sensitivity move events.
accel.shake.settings(<threshold>, <count>)	expression	Threshold (default 20) and count (default 10) settings defines sensitivity shake events.
button.green.down	boolean	True while the green button is being pressed.
button.green.pulse	boolean	True for a single cycle if the green button is pressed.
button.green.short	boolean	True for a single cycle if the green button is pressed shortly.
button.green.long	boolean	True while the green button is pressed for more than half a second.
button.red.hold	boolean	True while the red button is pressed for more than two seconds.
button.red.down	boolean	True while the red button is being pressed.

SwinxsTalk Reference Card – Expressions & Statements

Syntax	Type	Meaning
button.red.short	boolean	True for a single cycle if the red button is pressed shortly.
button.red.long	boolean	True while the red button is pressed for more than half a second.
button.red.hold	boolean	True while the red button is pressed for more then 2 secs.
button.white.down	boolean	True while the white button is being pressed.
button.white.short	boolean	True for a single cycle if the white button is pressed shortly.
button.white.long	boolean	True while the white button is pressed for more than half a second.
button.white.hold	boolean	True while the white button is pressed for more than two seconds.
button.volume.on	statement	Enables changing the volume via the white-short button.
button.volume.off	statement	Disables changing the volume via the white-short button.
button.power.on	statement	Enables turning off Swinxs via the white-short button.
button.power.off	statement	Prevent turning of Swinxs via the white-long button.
button.menu.on	statement	Enables the game menu via the red-long button.
button.menu.off	statement	Disables the game menu via the red-long button.
shuffle.on(<expr>)	statement	Initializes the shuffle at the next cycle with a list of length <expr>. Each item contains a unique random value between 0 and <expr> - 1.
shuffle.get	expression	The next item in the shuffle list. Only the next cycle, you will get a new shuffle item.
shuffle.rewind	statement	Rewinds the shuffle list to the beginning.
shuffle.off	statement	You *must* turn the shuffle off for one cycle before you can start a new one using shuffle.on(..).
shuffle.end	boolean	True when the last element of the shuffle list is read.
game.start(<expr>)	statement	Load and run a new game file, replacing the current one.
game.finish.score	statement	Exit your game by telling the scores (which are also saved to disk), and then:
game.finish.intro	statement	Exit your game by congratulate the players, and then:
game.finish.button	statement	Exit your game by telling to press the green or red button, and then:
game.finish.wait	statement	Exit your game by waiting for the green (play your game again) or the red button (leave your game and play another).
game.debug.log(<expr>)	statement	Show the value of <expr> in the SwinxsEmulator status log.
game.debug.break	statement	Within the SwinxsEmulator, stops executing your game and enters step-mode.

Syntax	Type	Meaning
memory.word.saveall	statement	Save all word variables and arrays to persistent memory.
memory.word.loadall	statement	Load a previously saved memory dump.
power.off	statement	Turn off Swinxs.
power.sleep.timeout (<expr>)	statement	Set the timeout after which Swinxs falls asleep (in tenths of seconds).
power.battery.low	boolean	True when Swinxs is on low power.
swinxs.sct(<string>)	statement boolean expression	Embed the string as SwinxsCoreTalk code, e.g. sct("MVI 10").
swinxs.capability (<string>)	boolean expression	Test whether the installed firmware supports a feature.

Syntax	Meaning
{ <statements> }	Groups semicolon separated statements together
if (<expr>) <statement>	Execute a <statement> if the boolean <expr> is true.

if (<expr>)
  <statement>
else
  <statement>
	The else-part is optionally.

<var> = <expr>	Assign an expression to a word or boolean variable 'var'.
<var>[<expr>] = <expr>	Assign an expression to an element of the array 'var'. Indexes should run from 0 up to, but not including, the size of the array.
<var>++	Increment word <variable> by one.
<var>--	Decrement word <variable> by one.